Examen 25 de julio de 2018

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

Formato

- Indique su nombre completo y número de cédula en cada hoja (no se corregirán las hojas sin nombre). Numere todas las hojas e indique la cantidad total de hojas en la primera.
- Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva.
- Si se entregan varias versiones de un problema solo se corregirá el primero de ellos.

Dudas

- Sólo se contestarán dudas de letra.
- No se aceptarán dudas en los últimos 30 minutos del examen.

Material

- El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del examen, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.

Aprobación

- Para aprobar el examen se debe tener un mínimo de 60 puntos.

Finalización

- El examen dura 4 horas.
- Al momento de finalizar el examen no se podrá escribir absolutamente nada en las hojas, debiéndose parar e ir a la fila de entrega. Identificar cada una de las hojas con nombre, cédula y numeración forma parte de la duración del examen.

Problema 1 (32 puntos) [4 puntos c/u]

Conteste justificando brevemente cada una de las preguntas.

1. Describa brevemente las funcionalidades más importantes que brinda el sistema de archivos virtual.
2. Describa tres métodos para la asignación de espacio de disco. Mencione una desventaja de cada uno de ellos.
3. Describa el funcionamiento de las interfaces de aplicación (system calls) de E/S de tipo no bloqueante y de tipo asincrónicas. Suponga que se tiene una llamada al sistema que soporta ambas interfaces ¿En que caso utilizaría una interfaz no bloqueante y en que caso una asincrónica?
4. Describa que es y como funciona la tabla de páginas oculta de un hipervisor. ¿Qué problemas pueden surgir con la MMU al utilizar una tabla de este tipo?
5. Explique como se manipula el stack para realizar un ataque de tipo stackoverflow. Describa un método para prevenirlo.
6. Describa las técnicas de E/S mapeada a memoria y E/S directa. Mencione una ventaja de cada una de estas técnicas.
7. Describa como funciona la protección de CPU. ¿Por qué no se puede asegurar la protección de CPU solamente con un algoritmo de planificación adecuado?
8. Describa la interfaz que brinda un sistema operativo para exponer sus servicios a las aplicaciones de usuario. Describa los pasos que sigue una invocación a un servicio de esta interfaz.

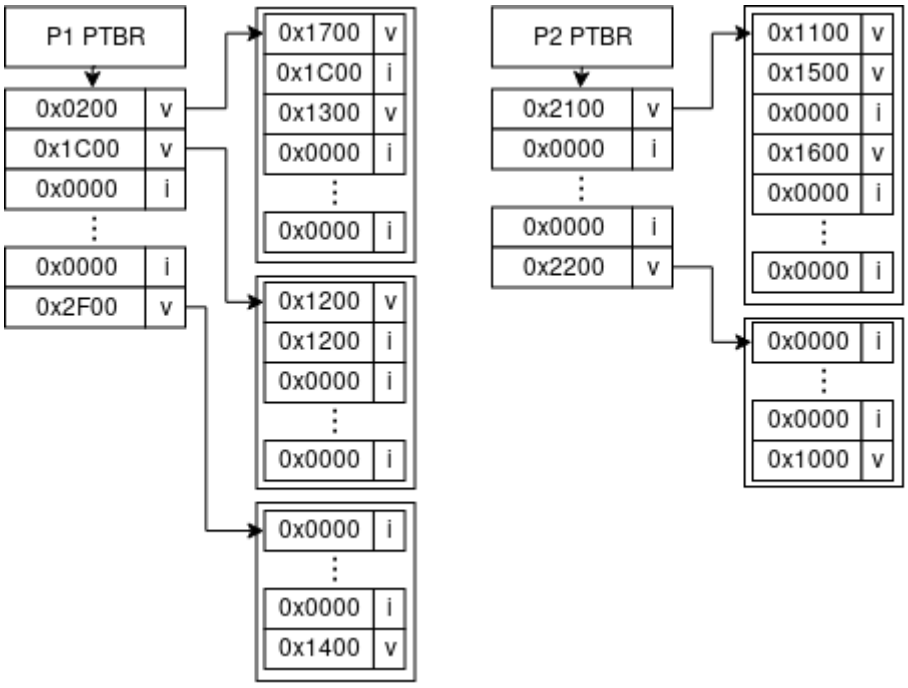
Problema 2 (34 puntos) [2, 5, 2, 5, 16 y 4 puntos]

Se tiene una sistema mono-procesador que ejecuta un SO con: planificación round-robin con un quatum de 2 unidades de tiempo, direcciones virtuales de 16 bits, paginación bajo demanda pura con páginas de 256 bytes, tablas de páginas de dos niveles donde cada tabla ocupa una página y tiene entradas de 16 bytes, y algoritmo de reemplazo de segunda chance.
El sistema cuenta solamente con 2 KB de memoria física para todos los procesos de usuario. Estos procesos pueden ejecutar tres acciones: accesos a memoria (M), operaciones con registros de CPU (R), y systemcalls de acceso a dispositivos de E/S (SC). Ejecutar una instrucción M o R requiere siempre 1 unidad de tiempo (sin importar si la operación es exitosa o no). Las operaciones de acceso a memoria (M) reciben como parámetro la dirección virtual a la cual se accede, ej. M(0xFFFF). Las operaciones con registros (R) no reciben parámetros. Las systemcalls de E/S (SC) tienen un consumo despreciable de procesador, requieren 4 unidades de tiempo para terminar la E/S y reciben como parámetro el identificador del dispositivo, ej.: SC(5).

Suponga se tienen dos procesos de usuario en ejecución, P1 y P2, que realizaron los siguientes accesos a memoria (ordenados de más reciente a más antiguo):

Proceso	P1	P1	P1	P2	P1	P2	P2	P2
Dirección	0xFF02	0x02AB	0x0013	0xFF11	0x1000	0x00FF	0x0144	0x03C4

Luego de estos accesos, el estado de sus tablas de páginas es:



Sabiendo que el espacio de direcciones virtuales de P1 y P2 esta compuesto por los rangos:
P1: [0x0000-0x02FF], [0x1000-0x12FF] y [0xFF00-0xFFFF].
P2: [0x0000-0x03FF] y [0xFF00-0xFFFF].

Se pide contestar:

1. ¿Cuántos bits son utilizados para determinar el desplazamiento y cuántos para determinar las entradas en la tabla de primer y segundo nivel? Justifique.
2. Para cada proceso: ¿Cuál es el tamaño de su memoria residente y cuanta memoria ocupa su stack? Justifique.
3. ¿Qué sucede si P1 realiza la operación M(0x1311)? Justifique.

4. Describa el funcionamiento del algoritmo de segunda chance.
5. Sea la siguiente secuencia de operaciones de P1 y P2, y suponiendo que en el primer instante de tiempo ambos procesos están listos y se la asigna el procesador al proceso P1:

P1	M(0x0014)	M(0x0100)	R()	M(0x112A)	R()	M(0x122A)	M(0xFF03)	R()
P2	M(0x0100)	SC(13)	M(0x0100)	M(0xFF11)	R()			

   a. Muestre un esquema que para cada instante de tiempo indique el estado de cada proceso y la operación que se esta ejecutando. En caso de ser una operación de acceso a memoria, mostrar el estado del algoritmo de reemplazo luego de la operación junto con el marco asignado a cada página válida.
   b. Determine cuántos fallos de página se produjeron. Justifique.

Solución:

1. Las páginas tienen un tamaño de 256 bytes, por lo tanto se requieren 8 bits para referenciarlas en su totalidad. Eso nos deja 8 bits para el direccionamiento de las páginas. Como cada tabla de página ocupa una página (2^8 bytes) con entradas de 16 bits, entonces se tiene que en total habrán $2^8/2^4=2^4$ entradas por tabla requiriendose 4 bits para direccionarlas. En resumen serán necesarios 4 bit por nivel de tabla de página y 8 bits para el desplazamiento.

2. Ambos procesos tienen 4 páginas cargadas en marcos de memoria, por lo tanto su memoria residente es de 1 KB cada uno.

3. Se produce un page fault y se termina la ejecución de P1 porque 0x1311 no pertenece al espacio de direcciones virtuales de P1.

4. Cada página tiene una marca que indica si fue referenciada (bit de referencia) luego de ser cargada en memoria (es decir que las páginas son cargadas con el bit de referencia en 0). En el momento de reemplazo, se verifica el bit de referencia. Si está prendido, a la página se le da una segunda chance y es puesta al final de la cola. Luego se continua con la siguiente página que está al principio de la cola. Si está apagado, es seleccionada para ser reemplazada

5.

P1	E/M(0x0014)	E/M(0x0100)	L/1	R/R()	E/M(0x112A)	E/R()	E/M(0x122A)	L/1	L/1	E/M(0xFF03)	E/R()	T	T	
P2	L/1	L/1	E/M(0x0100)	B/SC(13)	B	B	B	E/M(0x0100)	E/M(0xFF11)	L/1	L/1	E/R()	T	
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**

Para mostrar la cola del algoritmo de reemplazo se utilizan tuplas con el siguiente formato: (proceso, dir. página, dir. marco, bit de referencia). Para ahorrar espacio se presentan solamente los primeros 8 bits de las direcciones de página y marco, los restantes 8 bits corresponden a ceros.

Inicial: (p2,03,16,0), (p2,01,15,0), (p2,00,11,0), (p1,10,12,0), (p2,FF,10,0), (p1,00,17,0), (p1,02,13,0), (p1,FF,14,0)

t=0: (p2,03,16,0), (p2,01,15,0), (p2,00,11,0), (p1,10,12,0), (p2,FF,10,0), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,0)

t=1: (p2,01,15,0), (p2,00,11,0), (p1,10,12,0), (p2,FF,10,0), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,0), (p1,01,16,0) *

t=2: (p2,01,15,1), (p2,00,11,0), (p1,10,12,0), (p2,FF,10,0), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,0), (p1,01,16,0)

t=3: sin cambio

t=4: (p1,10,12,0), (p2,FF,10,0), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,0), (p1,01,16,0), (p2,01,15,0), (p1,11,11,0) *

t=5: sin cambio

t=6: (p2,FF,10,0), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,0), (p1,01,16,0), (p2,01,15,0), (p1,11,11,0), (p1,12,12,0) *

t=7: (p2,FF,10,0), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,0), (p1,01,16,0), (p2,01,15,1), (p1,11,11,0), (p1,12,12,0)

t=8: (p2,FF,10,1), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,0), (p1,01,16,0), (p2,01,15,1), (p1,11,11,0), (p1,12,12,0)

t=9: (p2,FF,10,1), (p1,00,17,1), (p1,02,13,0), (p1,FF,14,1), (p1,01,16,0), (p2,01,15,1), (p1,11,11,0), (p1,12,12,0)

t=10-12: sin cambio

Se muestra con * los instantes en que ocurren fallos de página. En total se producen 3 fallos.

Problema 3 (34 puntos)

Se desea modelar en ADA una panadería en la que trabajan 5 amasadores los cuales elaboran masa para hacer postres. Cada uno tiene una máquina amasadora la cual puede usar una o dos paletas. Se dispone de 5 paletas de amasado compartidas para todos los amasadores. La masa debe amasarse con una paleta a menos que la consistencia no sea la correcta en cuyo caso debe usarse una segunda paleta en conjunto con la primera para terminar el trabajo.

Luego de terminar de amasar el trabajador debe enjuagar la o las paletas usadas y liberarlas para quien las quiera usar. A continuación pondrá la masa en el horno y tomará una nueva mezcla para amasar.

En la panadería hay 5 supervisores de paletas, cada uno de los cuales verifica el estado de una paleta específica. Cuando el supervisor está esperando para supervisar una paleta los trabajadores no podrán agarrarla para ser usada.

Se dispone de los siguientes procedimientos:

- amasar(): boolean
  Ejecutada por los amasadores. Retorna si la consistencia de la masa es la correcta. La consistencia siempre es correcta cuando se amasa utilizando 2 paletas.
- colocarPaleta(integer paleta)
  Ejecutada por los amasadores para colocar la paleta en la amasadora.
- enjuagarPaleta(integer paleta)
  Ejecutada por los amasadores para limpiar la paleta y dejarla disponible para el resto.
- hornear()
  Ejecutada por los amasadores para poner la masa terminada en el horno.
- agarrarNuevaMasa()
  Ejecutada por los amasadores para tomar una nueva masa.
- supervisarPaleta(integer paleta)
  Ejecutada por los supervisores.
- otrasTareas
  Ejecutada por el supervisor luego de supervisar una paleta.

Solución:

```
task type amasador is
end amasador;


task body amasador is
    paleta1, paleta2 : integer;
begin
    loop
        agarrarNuevaMasa();
        panaderia.damePaleta(paleta1);
        colocarPaleta(paleta1);
        if not amasar() then
            panaderia.dameOtraPaleta(paleta2);
            colocarPaleta(paleta2);
            amasar();
```

```
                        enjuagarPaleta(paleta2);
                        panaderia.retornarPaleta(paleta2);
                end if;
                enjuagarPaleta(paleta1);
                panaderia.retornarPaleta(paleta1);
                hornear();
        end loop;
end amasador;

task type supervisor is
        entry config(p : integer);
        entry paletaLista();
end supervisor;

task body supervisor is
        paleta : integer;
        ok : boolean;
begin
        accept config(p : integer)
                paleta:=p;
        end
        loop
                panaderia.supervisarPaleta(paleta, ok);
                if not ok then
                        accept paletaLista();
                end if;
                supervisarPaleta(paleta);
                panaderia.retornarPaleta(paleta);
                otrasTareas();
        end loop;
end supervisor;

task panaderia is
        entry damePaleta(p : out integer);
        entry dameOtraPaleta(p : out integer);
        entry supervisarPaleta(paleta : integer, ok : out boolean);
        entry retornarPaleta(paleta : integer);
```

```
end panaderia;

task body panaderia is
	paletas : array(1..5) of boolean;
	esperando : array(1..5) of boolean;
	cant_paletas, i, actual, paleta : integer;
begin
	for i in range 1..5 loop
		paletas[i]:=true;
		esperando[i]:=false;
		supervisores[i].config(i);
	end loop;
	cant_paletas:=5;
	loop
		if cant_paletas > 0 then
			paleta := -1;
			i := 1;
			while paleta = -1 loop
				if paletas[i] then
					paletas[i]:=false;
					paleta:=i;
				end if;
				i:=i+1;
			end loop;
		end if;

		select
			when (cant_paletas >= 2) and (supervisarPaleta'count = 0) =>
				accept damePaleta(p : out integer) do
					p := paleta;
				end damePaleta;
				cant_paletas:=cant_paletas-1;
			or
			when (cant_paletas >= 1) and (supervisarPaleta'count = 0) =>
				accept dameOtraPaleta(p : out integer) do
					p := paleta;
				end damePaleta;
```

```
                        cant_paletas:=cant_paletas-1;
                or
                        accept supervisarPaleta(paleta : integer, ok : out boolean) do
                                ok:=paletas[paleta];
                                actual:=paleta;
                        end supervisarPaleta;
                        if paletas[actual] then
                                paletas[actual]:=false;
                                cant_paletas:=cant_paletas-1;
                        else
                                esperando[actual]:=true;
                        end if;
                or
                        accept retornarPaleta(paleta : integer) do
                                actual:=paleta;
                        end retornarPaleta;
                        if esperando[actual] then
                                esperando[actual]:=false;
                                supervisores[actual].paletaLista();
                        else
                                cant_paletas:=cant_paletas+1;
                                paletas[actual]:=true;
                        end if;
        end select;
    end loop;
end panaderia;

amasadores : array(1..5) of amasador;
supervisores : array(1..5) of supervisor;
```